BÀI THỰC HÀNH SỐ 4

Sử dụng bộ dữ liệu có sẵn Digikala (Tên một hệ thống siêu thị lớn ở khu vực châu Á). Bộ dữ liệu này bao gồm các bức ảnh chụp các mặt hàng trên trang web với các màu sắc khác nhau. Trong bài thực hành này sinh viên sử dụng lược đồ histogram của kênh màu H trong HSV để làm vector nhận dạng. Hãy trích xuất đặc trưng cho tập ảnh, với mỗi ảnh thu được một vector histogram (H) và sử dụng nó để nhận dạng.

- Hàm crop_image để lược bỏ những vùng trắng xung quanh ảnh, tăng độ chính xác khi áp dụng thuật toán.

```
def crop_image(img):
    """
    This is a function that crops extra white background
    around product.
    Src:
        https://stackoverflow.com/questions/64046602/how-can-i-crop-an-ob
    """
    mask = img!=255
    mask = mask.any(2)
    mask0,mask1 = mask.any(0),mask.any(1)
    colstart, colend = mask0.argmax(), len(mask0)-mask0[::-1].argmax()+1
    rowstart, rowend = mask1.argmax(), len(mask1)-mask1[::-1].argmax()+1
    return img[rowstart:rowend, colstart:colend]
```

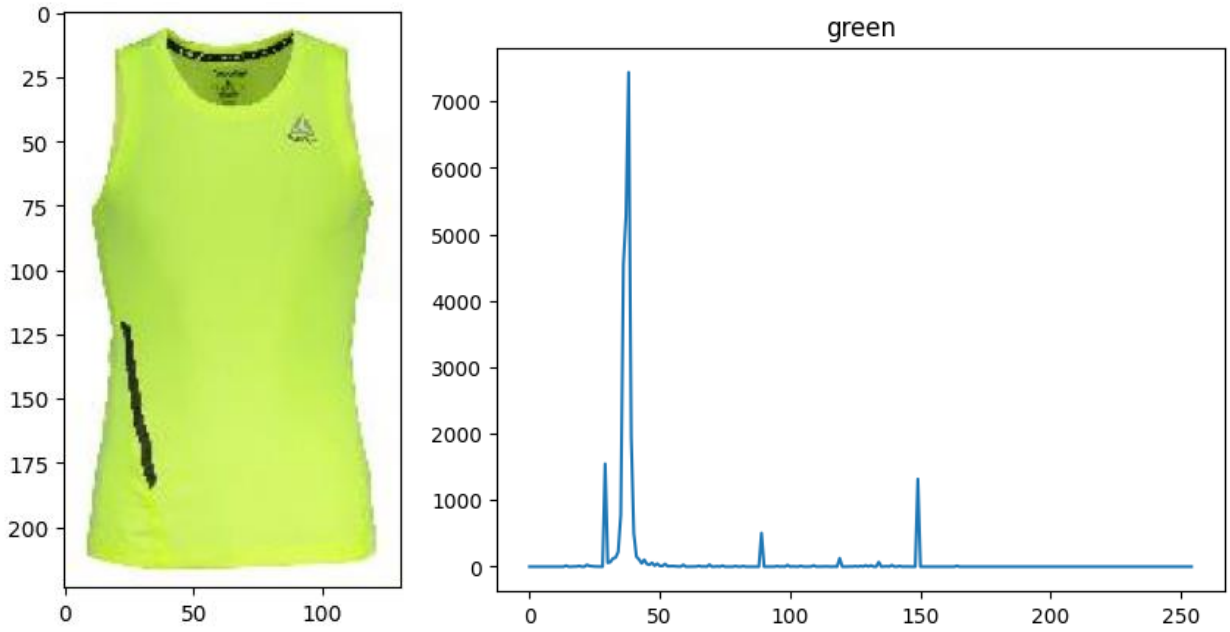
- Hàm hsv_histogram để tính histogram của hình ảnh.

```
def hsv_histogram(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
    h = hsv[..., 0]
    return np.bincount(h.ravel(), minlength=256)
```

- Lấy tất cả các đường dẫn ảnh:

```
root = 'digikala/train'
images = list(Path(root).rglob('*.jp*g'))
```

- Vẽ biểu đồ histogram của ảnh ngẫu nhiên:



- Gán nhãn cho từng màu:

```
{'black': 0,
 'blue': 1,
 'brown': 2,
 'green': 3,
 'grey': 4,
 'orange': 5,
 'pink': 6,
 'purple': 7,
 'red': 8,
 'silver': 9,
 'white': 10,
 'yellow': 11}
```

- Tính vector histogram cho từng ảnh:

```
for i, path in enumerate(paths):
    imgBGR = cv2.imread(path.as_posix())
    imgRGB = cv2.cvtColor(imgBGR, cv2.COLOR_BGR2RGB)
    label = labels[path.parent.stem]
    img = crop_image(imgRGB)
    hist = hsv_histogram(img)
    df = pd.concat([df, pd.DataFrame(hist[1:]).T], ignore_index=True)
    col_labels.append(label) # append labels to insert after the loop

df['labels'] = col_labels # thêm nhãn vào dữ liệu
```

- Dữ liệu thu được:

```
df.head()
```

✓ 0.0s

	0	1	2	3	4	5	6	7	8	9	...	246	247	248	249	250	251	252	253	254	labels
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	36	40	63	50	120	185	40	168	73	76	...	0	0	0	0	0	0	0	0	0	0
4	0	0	41	1	50	3944	0	136	1	14	...	0	0	0	0	0	0	0	0	0	0

- Chia dữ liệu thành 2 tập train và test với tỷ lệ 80/20.
- Áp dụng thuật toán KNN và SVM thu được độ chính xác như sau:

```
SVC
0.5897435897435898
KNeighborsClassifier
0.5913461538461539
```